I. Agudo, et al., "Delegation Perspective of Practical Authorization Schemes", *Fifth International Network Conference (INC05)*, pp. 157-164, 2005.
NICS Lab. Publications: https://www.nics.uma.es/publications

Delegation Perspective of Practical Authorization Schemes

Isaac Agudo[1], Javier Lopez[1], Jose A. Montenegro[1], Eiji Okamoto[2] and Ed Dawson[3]

[1]Computer Science Department, E.T.S. Ingenieria Informatica
University of Malaga, Spain
email:{isaac,jlm,monte}@lcc.uma.es
[2]Institute of Information Sciences and Electronics
University of Tsukuba, Japan
email:okamoto@is.tsukuba.ac.jp
[3]Information Security Institute
Queensland Univ. of Technology, Australia
email:e.dawson@qut.edu.au

Abstract

Different authorization schemes for Internet applications have been proposed during the last years as solutions for the distributed authorization problem. Because delegation is a concept derived from authorization, this paper studies and put into perspective the delegation implications, issues and concepts that are derived from a number of those authorization schemes. For our study, we have selected a group of authorization schemes based on two issues: their support from international bodies, and the practicality to be deployed and used in real-world Internet applications.

Keywords

Authorization, Certificate, Delegation, Trust Relation

1 Introduction

As it is widely known, computer and network security are related to the Internet more than ever before. As a consequence, the use of Internet has produced variations in security software. A number of these changes focus on the way users are authenticated by Internet applications and how their rights and privileges are managed.

Therefore, one of the most controversial security services is *Access Control*. Lampson defines access control as the composition of two services, *Authentication* and *Authorization* (Lampson, 2004). Internet applications require distributed solutions for the access control service. Accordingly, authentication and authorization services need to be distributed too.

In order to achieve a real scalable distributed authorization solution, the *Delegation* service needs to be strongly considered. Delegation is quite a complex concept, both from the theoretical point of view and from the practical point of view. In this sense, the implementation of an appropriate delegation service is becoming a cornerstone of Internet applications since a few years ago.

Because delegation is a concept derived from authorization, this paper aims and put into perspective the delegation implications, issues and concepts that are derived from a selected group of authorization schemes that have been proposed during the last years as solutions for the distributed authorization problem.

For this work, the group of selected authorization solutions are *KeyNote*, as an evolution of *PolicyMaker, SDSI/SPKI* and *Privilege Management Infrastructures* (PMI). The reason for selecting these schemes is twofold. On the one hand, they are supported by international bodies. This is the case of IETF, that has supported the two first ones through several RFCs. It is also the case of ITU-T, that has proposed and supported the PMIs. It is important to note that PMIs have been supported by the IETF too. On the other hand, selected schemes can be considered as practical solutions, so they can be deployed in the Internet more easily than other more specific approaches based on formalisms (graph theory or logic programming), like (Li et al., 2002), (Li et al., 2003), (Varadharajan et al., 2003), (Ruan and Varadharajan, 2004).

The rest of the paper is structured as following. Section 2 covers, PolicyMaker and KeyNote solutions. Section 3 analyzes the SPKI/SDSI solution in global, but also the SDSI solution in particular. Section 5 elaborates on ITU-T PMIs. Finally, section 6 concludes the paper.

2 PolicyMaker and Keynote

Blaze, Feigenbaum and Lacy introduced in (Blaze et al., 1996) the notion of *Trust Management*. In that original work they proposed the PolicyMaker scheme as a solution for trust management purposes.

PolicyMaker is a general and powerful solution that allows the use of any programming language to encode the nature of the authority being granted as well as the entities to whom it is being granted. It addresses the authorization problem directly, without considering two different phases (one for authentication and another for access control).

PolicyMaker encodes trust in assertions. They are represented as pairs (f, s), where s is the issuer of the statement, and f is a program.

Additionally, PolicyMaker introduces two different types of assertions: *certificates* and *policies*. The main difference between them is the value of the *Source* field. To be more precise, the value is a key for the first one (certificates), and a label for the second one (policies).

It is important to note that, in PolicyMaker, negative credentials are not allowed. Therefore, trust is monotonic; that is, each policy statement or credential can only increase the capabilities granted by others. Moreover, trust is also transitive. This means that if *Alice* trusts *Bob* and, extensively, *Bob* trusts emphCarol, then *Alice* trusts *Carol*.

In other words, all authorizations are delegable. Indeed, delegation is implicit in PolicyMaker; thus, it is not possible to restrict delegation capabilities. This is the reason why delegation is uncontrolled in PolicyMaker.

KeyNote (Blaze et al., 1999) has been proposed and designed to improve two main aspects of PolicyMaker: to achieve standardization and to facilitate its integration into applications.

Keynote uses a specific assertion language, that is flexible enough to handle the security policies of different applications. Assertions delegate the authorization to perform operations to other principals. As PolicyMaker, KeyNote considers two types of assertions. Also, as in PolicyMaker, these two types of assertions are called *policies* and *credentials*, respectively:

- Policies. This type of assertions do not need to be signed because they are locally trusted. They do not contain the corresponding *Issuer* of PolicyMaker.
- Credentials. This type of assertions delegate authorization from the issuer of the credential, or *Authorizer*, to some subjects or *Licensees* (see later for details).

Assertions are valid or not valid depending on *action attributes*, which are attribute/value pairs like `resouce == "database"` or `access == "read"`

KeyNote assertions are composed of five fields:

- `Authorizer`. If the assertion is a credential, then this field encodes the issuer of that credential. However, if the assertion is a policy, then this field contains the keyword **POLICY**.
- `Licensees`. It specifies the principal or principals to which the authority is delegated. It can be a single principal or a conjunction, disjunction or threshold of principals.
- `Comment`. It is a comment for the assertion.
- `Conditions`. It corresponds to the ”program” concept of PolicyMaker, and consists of tests on action attributes. Logical operators are used in order to combine them.
- `Signature`. It is the signature of the assertion. This field is not necessary for policies, only for credentials.

Further description on how KeyNote uses cryptographic keys and signatures can be found in (Blaze et al., 2000).

Figure 1 shows an example of assertion. It states that an RSA key 12345678 authorizes the DSA keys $abcd1234$ and $1234abcd$ for read and write access on the database.

```
KeyNote-Version: 2
Authorizer: "rsa-hex:12345678"
Licensees: "dsa-hex:abcd1234" || "dsa-hex:1234abcd"
Comment: Authorizer delegates read and write access to
         either of the licensees
Conditions: (resource == "database" &&
            (access == "read") || (access == "write"))
Signature: "sig-rsa-md5-hex:00001234"
```

Figure 1: KeyNote assertion

Given a set of action attributes, an assertion graph is a directed graph with vertex corresponding to principals. An arc exists from principal A to principal B if an assertion exists where the

Authorizer field corresponds with A, the *Licensees* field corresponds with B and the predicate encoded in the *Conditions* field holds for the given set of action attributes. A principal is authorized, under a given set of action attributes, if the associated graph contains a path from a policy to the principal. We conclude that all authorized principals are allowed to re-delegate their authorizations. Thus, there is no restriction on delegation.

3 SDSI/SPKI

This solution is an unification of two similar proposals, SDSI (Simple Distributed Security Infrastructure) and SPKI (Simple Public Key Infrastructure). SPKI was proposed by the IETF working group and, in particular, by Carl Ellison (Ellison et al., 1996). SDSI was an alternative design for a public-key infrastructure to X.509, designed by Ronald L. Rivest and Butler Lampson (Rivest and Lampson, 1996).

The $SPKI/SDSI$ certificate format is the result of the SPKI Working Group of the IETF (Ellison, 1999). The main feature of SDSI/SPKI is that its design provides a simple public key infrastructure which uses linked local name spaces rather than a global, hierarchical one. All entities are considered analogous; hence, every principal can produce signed statements.

The data format chosen for SPKI/SDSI is *S-expression*. This is a LISP-like parenthesized expression with the limitations that empty lists are not allowed and the first element in any S-expression must be a string, called the ”type” of the expression.

In this section, we detail the SDSI solution and the integrated solution SDSI/SPKI, as the development of the SPKI solution is similar to the integrated solution. The subsections detail the certificates of each proposal and explain how the delegation is implemented.

3.1 SDSI

SDSI establishes four types of certificates: $Name/Value$, $Membership$, $Autocert$ and $Delegation$.

Name/Value Certificates: These certificates are used to bind principals to local names. Every certificate must be signed by the issuer, using his/her public key (figure 2).

Membership Certificates: These are certificates that give to principals the membership to a particular SDSI group.

Autocert Certificates: These are self-certificates, a special kind of certificate. Every SDSI principal is required to have an Autocert (figure 3).

Delegation Certificates: These certificates are the mechanisms for implementing the Delegation in SDSI, (figure 3). SDSI provides two types of delegation, based on the structure of the delegation certificate:

(i) A user (issuer) can delegate to someone by adding that person as a member to a group issuer control. *A* issues a delegation certificate to *B*. Therefore, *B* will have the same privileges as the $group1$.

(ii) A user (issuer) can delegate to someone so that this person is able to sign objects of a certain type on the user's behalf. The ”certain type” is defined by using the template form.

```
(Cert:
   (Local-Name: user1 )
   (Value:
      (Principal:
        (Public-Key:
        (Algorithm: RSA-with-SHA1 )
        ......
        )))
        (Signed: ...))
```

Figure 2: Name-Value certificates

```
(Auto-Cert:                           (Delegation-Cert:
   (Local-Name: user1 )                  (Template: form )
     (Public-Key: ....)                    (Group: group1 )
     (Description: temporal user)          (Signed: ...))
     (Signed: ...))
```

Figure 3: Autocert and Delegation certificates

3.2 Integrated Solution, SPKI/SDSI

SPKI/SDSI unifies all types of SDSI certificates into one single type of structure. The SPKI/SDSI certificate contains at least an *Issuer* and a *Subject*, and it can contain validity conditions, authorization and delegation information. Therefore, there are three categories: ID (mapping <name,key>), Attribute (mapping <authorization,name>), and Authorization (mapping <authorization,key>).

Figure 4 details the relationship between key, name and authorization sentences and the three possible SDSI/SPKI certificate categories.

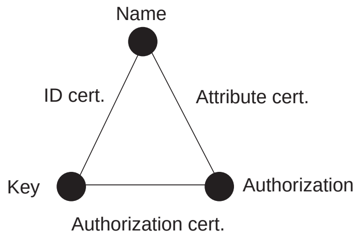


Figure 4: SPKI Certificate Types

The structure of Figure 5 represents the *ID certificate* and the *Authorization Certificate*. The *Attribute Certificate* has the same structure as Authorization Certificates.

```
(cert                                  (cert
    (issuer  <principal>)                  (issuer  <principal>)
    (subject <principal>)                  (subject <principal>)
    (valid   <valid>))                     (propagate)
                                           (tag <tag>)
                                           (valid ))
```

Figure 5: ID and Authorization Certificates

The field $propagate$ is the field used to perform the delegation. As it was desirable to limit the depth of delegation, initially, SPKI/SDSI had three options for controlling this: no control, boolean control and integer control. Actually these options have been reduced to boolean control only. In this way, if this field is true, the Subject is permitted by the Issuer to further propagate the authorization.

4 Privilege Management Infrastructure (PMI)

The last X.509 ITU-T Recommendation (ITU, 2000) introduces the concept of *Privilege Management Infrastructure* (PMI) as the framework for the extended use of *attribute certificates*. The Recommendation establishes four PMI models: (i) *General*, (ii) *Control*, (iii) *Roles* and (iv) *Delegation*. The first one can be considered as an abstract model, while the other ones can be considered as the models for implementation.

The PMI area inherits many concepts from the *Public Key Infrastructure* (PKI) area. In this sense, an *Attribute Authority* (AA) is the authority that assigns privileges (through attribute certificates) to users, and the *Source of Authorization* (SOA) is the root authority in the delegation chain. A typical PMI will contain a SOA, a number of AAs and a multiplicity of *end entities* (EE). (Farrel and Housley, 2002)

Figure 6 depicts the relation between the entities of a PMI in the Delegation Model. Initially, the Source of Authority assigns or delegates the privilege to Attribute Authorities. These can delegate the privileges to other AAs or to EEs.

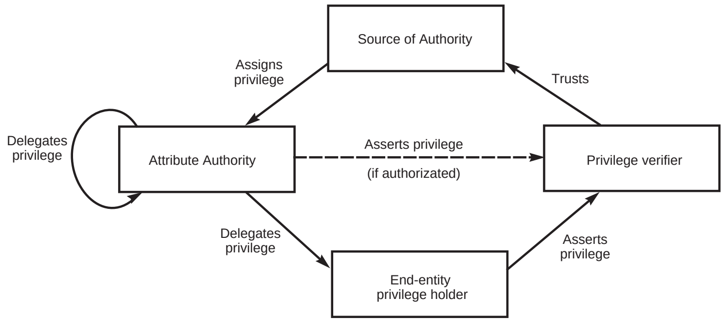
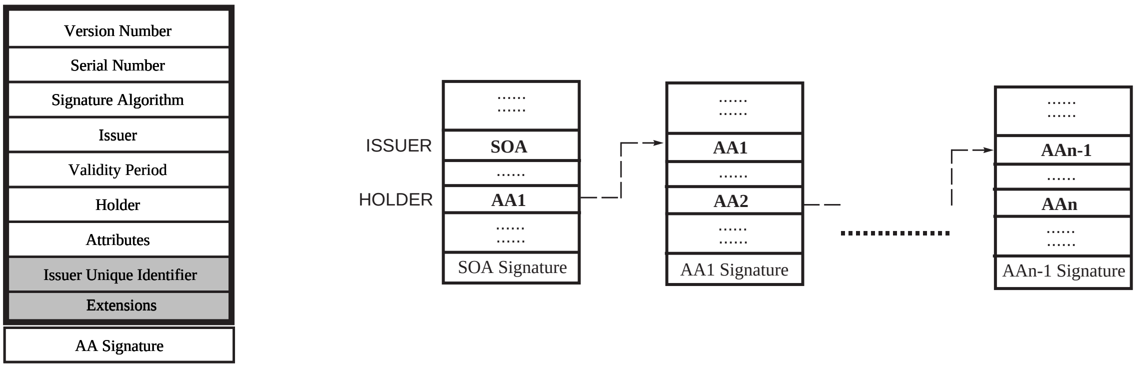


Figure 6: PMI Delegation Model

AAs and EEs can use their delegated privileges and present them to the *Privilege Verifier* (PV), that verifies the certification path to determine the validity of the privileges. The difference between AA and EE is that EE can not further delegate the privileges to other entities, becoming the leaves of the tree. The PV must trust the SOA in order to verify the certification path, as they may reside in different domains.

The mechanism (data structure) used to contain the delegation statement(s) is the attribute certificate. Figure 7 shows the structure of the attribute certificate, and how a delegation path is established through a chain of these certificates. The *Extensions* field is used by the authorities to include the delegation policy.



(a) X.509 Attribute Certificate (b) Delegation Path

Figure 7: Delegation Elements in PMI

5 Conclusions

Delegation is a goal needed to obtain a real scalable distributed authorization. However, the uncontrolled use of delegation statements can become a security threat because any user could improperly get the same privileges over a resource than the owner of that resource. Therefore, delegation solutions must include a mechanism to control the delegation and to produce appropriate authorizations statements. In this paper, our goal has been to study and put into perspective the delegation implications of a group of schemes that have been proposed as solutions for distributed authorization problems. In *PolicyMaker* and *Keynote* the delegation statement does not exist, and any authorization statement can be delegated again and again without any control. *SDSI* considers three different possibilities to control the delegation, although *SPKI* has reduced it to a boolean condition. Such a boolean parameter is only a modest mechanism to control the depth of the delegation. The *PMI* solution provides more complex mechanism to perform the delegation, allowing the possibility to use the extension fields of the attribute certificate to perform a controlled delegation.

Acknowledgements

This paper is an outcome of three Research Projects where the different co-authors have been involved. We thank the support of: (i) the European Commission through the Project UBISEC (IST-506926), (ii) the Japanese National Institute of Information and Communication Technology (NICT) through the International Research Project “Secure Privacy Infrastructure” and, (iii) the Spanish Ministry of Science and Technology through the project PRIVILEGE (TIC-2003-8184-C02). We also thank the Andalusian Regional Government for supporting the PhD work of the first author.

References

Blaze, M., Feigenbaum, J., Ioannidis, J. and Keromytis, A. 1999. “The KeyNote Trust-Management System Version 2.” *RFC 2704*.

Blaze, M., Feigenbaum, J. and Lacy, J. 1996. “Decentralized Trust Management.” In *IEEE Symposium on Security and Privacy*. IEEE Computer Society Press pp. 164–173.

Blaze, M., Ioannidis, J., and Keromytis, A. 2000. “DSA and RSA Key and Signature Encoding for the KeyNote Trust Management System.” *RFC 2792*.

Ellison, C., Frantz, B. and Lacy, J. 1996. “Simple public key certificate”. *Internet Draft* `http://world.std.com/~cme/spki.txt`

Ellison, C., Frantz, B., Lampson, B., Rivest, R., Thomas, B. and Ylonen, T. 1999. “SPKI Certificate Theory”, *RFC 2693*.

Farrell, S. and Housley, R. 2002. “An Internet Attribute Certificate Profile for Authorization”, *RFC 3281*.

ITU. 2000. *X509.Information technology Open systems interconnection The Directory: Public-key and attribute certificate frameworks*.

Lampson, B. 2004. “Computer security in the real world.” *IEEE Computer Society Press* 37(6):37,46.

Li, N., Mitchell, J.C. and Winsborough, W.H. 2002. “Design of a Role-Based Trust Management Framework.” In *Proceedings of the 2002 IEEE Symposium on Security and Privacy*. IEEE Computer Society Press pp. 114–130.

Li, N., Grosof, B. and Feigenbaum, J. 2003. “Delegation logic: A logic-based approach to distributed authorization.” *ACM Trans. Inf. Syst. Secur.* 6(1):128–171.

Rivest, R. and Lampson, B. 1996. “SDSI - A Simple Distributed Security Infrastructure.” *Working document*, Presented at CRYPTO ’96 Rumpsession.

Ruan, C. and Varadharajan, V. 2004. “A Weighted Graph Approach to Authorization Delegation and Conflict Resolution.” In *ACISP 2004*. Vol. 3108 of *Lecture Notes in Computer Science*, Springer pp. 402–413.

Varadharajan, V., Ruan, C. and Yan Zhang. 2003. “A Logic Model for Temporal Authorization Delegation with Negation.” In *6th International Information Security Conference, ISC*. Vol. 2851 of *Lecture Notes in Computer Science*, Springer pp. 310–324.